Table of Contents

* [Prometheus Installation Lab](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_prometheus_installation_lab)
* [1. Set Up Prometheus](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_set_up_prometheus)
* [2. Set Up Node Exporter](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_set_up_node_exporter)
* [3. Verify Prometheus Installation and Explore Capabilities](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_verify_prometheus_installation_and_explore_capabilities)
  + [3.1. Verify Prometheus Infrastructure](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_verify_prometheus_infrastructure)
  + [3.2. Explore Queries and Metrics](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_explore_queries_and_metrics)
* [4. Explore Grafana Dashboards](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_explore_grafana_dashboards)
  + [4.1. Deploy Grafana](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_deploy_grafana)
  + [4.2. Configure Grafana Data Source](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_configure_grafana_data_source)
  + [4.3. Configure Grafana Dashboard](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_configure_grafana_dashboard)
  + [4.4. Explore Grafana Dashboard](https://www.opentlc.com/labs/ocp_operations/03_01_Prometheus_Monitoring_Lab.html#_explore_grafana_dashboard)

Prometheus Installation Lab

Prometheus and its related components can be set up completely on top of OpenShift. A set of templates is provided.

1. Set Up Prometheus

1. Log in to your OpenShift Bastion host and switch to the **root** user.
2. Clone the repository containing all of the Prometheus setup code:
3. cd /root
4. git clone https://github.com/wkulhanek/openshift-prometheus
5. cd openshift-prometheus

git checkout 3.6

1. Make sure you are logged into OpenShift as a user with cluster-admin privileges. Usually this will be the **system:admin** user.

oc login -u system:admin

1. Create an OpenShift project to hold Prometheus and all of the related components:

oc new-project prometheus --display-name="Prometheus Monitoring"

1. Open port **1936** on the nodes so that Prometheus is able to monitor the routers:

ansible-playbook ./update\_firewall.yml

* + As of the release of OpenShift Container Platform 3.6, the OpenShift routers expose a metrics endpoint on port **1936**. This port must be open for Prometheus to be able to monitor the routers.

1. Find the router password:

oc set env dc router -n default --list|grep STATS\_PASSWORD|awk -F"=" '{print $2}'

* + The router requires basic authentication to scrape the metrics.

1. Instantiate the Prometheus template using the retrieved router password as a parameter:

oc new-app -f prometheus.yaml --param ROUTER\_PASSWORD=<Router Password>

1. While Prometheus is deploying, examine the Prometheus template in **prometheus.yaml**.
   * Prometheus runs under its own service account, and this service account is granted **cluster-reader** permission.
   * The Prometheus service includes annotations (**prometheus.io/scrape: "true"** and **prometheus.io/scheme: http**) that allow Prometheus to monitor itself.
   * Prometheus uses an EmptyDir to store data. This data will be lost upon Pod Restart. If persistent storage is required the Prometheus team recommends HostPath storage and not NFS backed storage.
   * There are two configuration maps to configure Prometheus:
     + The first configuration map (**prometheus**) contains the entire Prometheus configuration. It contains a section to point to rules for alerting.
       - Note how the configuration sets up the various **scrape\_config** sections to identify targets for Prometheus.
     + The second configuration map (**prometheus-rules**) contains the alerting rules. It is mounted in the location the first configuration map points to.
     + Based on your requirements, you could customize these configurations.

2. Set Up Node Exporter

Prometheus is able to self-discover a wealth of information. For Prometheus to display node and Docker metrics, it is necessary to deploy the *node exporter*. The node exporter runs as a [DaemonSet](https://docs.openshift.com/container-platform/latest/dev_guide/daemonsets.html) to guarantee that each node runs at least one copy of the application.

|  |  |
| --- | --- |
|  | To deploy a DaemonSet and set the correct permissions for the pods, your OpenShift user needs **cluster-admin**authorization. |

1. Switch to the **node-exporter** directory and run an Ansible Playbook to open port **9100** on every node in the cluster:
2. cd /root/openshift-prometheus/node-exporter

ansible-playbook -i /etc/ansible/hosts update\_firewall.yml

1. Add the default service account in the **prometheus** project to the **privileged** SCC:

oc adm policy add-scc-to-user privileged system:serviceaccount:prometheus:default

* + Because the node exporter accesses resources from each host, the service account used to run the pod must be granted elevated access.

1. Set an empty node selector for the **prometheus** project:

oc annotate namespace prometheus openshift.io/node-selector=""

* + Most OPENTLC environments include a default project node selector. This default node selector prevents the DaemonSet from placing pods on masters and infranodes.

1. Deploy the DaemonSet:
2. oc project prometheus

oc new-app -f node-exporter.yaml

1. Switch to the directory containing the **dockerinfo** scripts and run an Ansible Playbook to distribute the script to all of the nodes and set up a **cron** job to execute the script at regular intervals:
2. cd /root/openshift-prometheus/node-exporter/dockerinfo

ansible-playbook -i /etc/ansible/hosts install\_dockerinfo.yml

* + While the node exporter already provides a wealth of information about each node, if you want specific information about Docker and the Docker volume group, the repository contains a few scripts that run on each node to collect this information.

The Prometheus infrastructure setup is complete.

3. Verify Prometheus Installation and Explore Capabilities

3.1. Verify Prometheus Infrastructure

In this section, you verify that Prometheus is set up properly by navigating to the route created for Prometheus and running a few sample queries.

1. In your web browser, navigate to the **targets** page of your Prometheus server, replacing the route before **/targets** with your specific route—for example, [http://"prometheus-prometheus.apps.GUID.example.opentlc.com/targets"](http://prometheus-prometheus.apps.guid.example.opentlc.com/targets).
   * Next to each endpoint, expect to see the state listed as **UP** in green.
     + Some service endpoints may show as **Down** in red. That is OK for our environment. Typically these are routers that have been discovered by Prometheus - but where Prometheus was not able to authenticate to scrape metrics. We have a special config for our router at the bottom that includes the authentication necessary to connect to the router.
   * There are four kinds of endpoints at the moment. Kubernetes includes built-in support for Prometheus, and you see the **kubernetes-node-exporter** endpoints for all of your nodes. Expect the number to match the number of **kubernetes-nodes**endpoints:
2. Switch to the **Graph** tab to try a few Prometheus queries.
3. In the **Expression** text box, type the following and then press **Enter**:

node\_docker\_running\_containers

* + Expect to see the number of running Docker containers for each node:

1. Try a few node-specific queries, which come from the OpenShift node service itself:
   * **node\_memory\_MemTotal**
   * **node\_memory\_MemFree**
   * **node\_cpu**
2. Note the detailed CPU information returned.

|  |  |
| --- | --- |
|  | To limit the CPU information to just the **user** mode, you can limit the query to just the **node\_cpu {mode="user"}** label. |

1. Try queries for individual container metrics:
   * **container\_memory\_usage\_bytes**
   * **container\_cpu\_usage\_seconds\_total**
   * **container\_network\_receive\_bytes\_total**
2. Check the other metrics that the **dockerinfo** script provides to the node exporters:
   * **node\_docker\_volume\_size\_bytes**
   * **node\_docker\_volume\_data\_percent\_full**
   * **node\_docker\_volume\_meta\_percent\_full**

3.2. Explore Queries and Metrics

1. Explore the queries.
2. Examine all of the available metrics from the list box next to the **Execute** button.
3. Switch from **Console** to **Graph** to show simplified graphical results of a query.

4. Explore Grafana Dashboards

The volume of information in Prometheus can be overwhelming. It can be more useful to present users with beautiful, graphical dashboards. Fortunately, the open source dashboard software Grafana can display data from Prometheus as well as a number of other systems.

You can easily deploy Grafana alongside Prometheus.

4.1. Deploy Grafana

1. On your Bastion host, make sure you are in the **prometheus** project and create the Grafana deployment:
2. oc project prometheus
3. oc new-app -f https://raw.githubusercontent.com/wkulhanek/docker-openshift-grafana/3.6/grafana.yaml

oc get route grafana

1. When you know your Grafana route (usually similar to **grafana-prometheus.apps.GUID.example.opentlc.com**), navigate to it in your web browser.

4.2. Configure Grafana Data Source

1. Log in to Grafana using **admin** default account and **admin** as the password.
   * It is a good idea to change the password after this.
2. On the **Home** dashboard, click **Add data source**.
3. For the **Config** data source, enter the following values:
   * **Name**: **DS-Prometheus**
   * **Type**: **Prometheus**
   * HTTP Settings:
     + **Url**: **http://prometheus:9090**
     + **Access**: **proxy**
4. Click **Add**.
5. Click **Save & Test**.
   * Expect to see a message indicating that the data source is working.

4.3. Configure Grafana Dashboard

In the repository, you can find an example dashboard JSON file called **openshift-metrics-dashboard.json**. This dashboard depends on metrics being collected by Prometheus as configured in previous steps of the lab.

There are more example dashboards available at <https://grafana.com/dashboards?search=kubernetes>.

1. Download this dashboard JSON file to your computer (**NOT** the Bastion host):
2. cd $HOME

curl -O https://raw.githubusercontent.com/wkulhanek/docker-openshift-grafana/3.6/openshift-metrics-dashboard.json

1. In Grafana, select the icon at the top left and then select **Dashboards → Import**.
2. Either copy/paste the contents of the JSON file you just downloaded (make sure to keep the correct formatting) or click **Upload .json File** selecting the JSON file from your **$HOME** directory.
3. In the next dialog, enter **OpenShift Metrics** as the name and select the previously created **DS-Prometheus** data source for **Prometheus**.
4. Click **Import**.
   * Expect to see your dashboard with metrics coming from Prometheus:

4.4. Explore Grafana Dashboard

1. Experiment using the Grafana dashboard.
2. Investigate some of the queries behind the graphs.

|  |  |
| --- | --- |
|  | If you hover over a graph, a box appears at the top where you can click **Edit** to bring up the dashboard widget editor. |

1. Try importing pre-built Kubernetes dashboards from <https://grafana.com/dashboards?search=kubernetes>.
   * Some data that these dashboards expect may not be available in your specific Prometheus installation.

Build Version: be32934b41da1e34c27e21e79bdf57d30c732a3e : Last updated 2018-01-19 05:59:04 EST